Community Embeddings for Friend Suggestions

Anton I. N. Begehr
Graduate School of Business
National Research University Higher School of Economics
Moscow, Russia
a.begehr@fu-berlin.de

Prof. Dr. Petr Panfilov
Graduate School of Business
National Research University Higher School of Economics
Moscow, Russia
ppanfilov@hse.ru

***Abstract*—Graphs, such as social networks, emerge naturally from various real-world situations. Recently, graph embedding methods have gained traction in data science research. Recommender systems are used in a wide range of business applications and are essential for online, e-business models to survive and strive in the contemporary market. Using graph embeddings for recommendation tasks, have the possibility of improving upon recommender systems, because of data compression, their feature vector format, and sub-quadratic time complexity. Graph and community embeddings generated with ComE BGMM+VI are used to build a recommender system for friend suggestions. ComE BGMM+VI is an alteration of the community embeddings algorithm ComE. ComE BGMM+VI applies a Bayesian Gaussian mixture model and variational inference for community embedding and detection. Recommendations are evaluated by the top-N hit-rate over users with at least 50 friends. A friend suggestions recommender system with a top-10 leave-one-out hit-rate of 43.6% and run-time optimized 32.9% is presented.**

***Index Terms*—graph, embedding, community embedding, ComE, recommender systems, friend suggestions**

I. Introduction

Graphs, such as social networks, knowledge graphs, content-rating graphs, and communication networks, emerge naturally from various real-world situations. Analyzing these graphs leads to findings and understanding of the underlying structures, coherences, and dependencies. Recently, methods for embedding graph's nodes into lower-dimensional Euclidean spaces, called graph embeddings, have gained traction in multiple areas of data science research [1].

Due to the rapid growth of the internet and data accumulation, recommender systems are essential for e-business and online business models to survive and strive in the contemporary market [2]. Modern recommender systems need to take into account the huge amounts of user data generated at all times in big data systems around the world and improve recommendations instead of failing under the thrust of big data overload.

Utilizing graph embeddings for recommendation tasks, has recently gained research traction [3, 4, 5, 6]. The advantages of graph embeddings include data compression and the Euclidean feature vector format [7]. Given these advantages and provided competitive results, graph embeddings have the possibility of greatly improving upon graph-based use-cases like recommender systems.

Community Embeddings, in addition to embedding a graph's nodes through first- and second-order proximity, also preserve higher-order proximity by embedding clusters present in the graph data. The graph and community embedding algorithm ComE aims to preserve first-, second- and higher-order proximity by embedding a graph's nodes and communities [8].

This work specifically examines community embeddings for friend suggestion recommender systems and evaluates recommendations on social network graph data for the use-case of friend suggestions. Graph and community embeddings generated with ComE BGMM+VI are used to develop a friend suggestions recommender system based on the shortest distances between nodes in the embedding. Recommendations are evaluated by the top-N recommendations hit-rate of test edges. A friend suggestions recommender system with a top-10 leave-one-out hit-rate of 43.6% and run-time optimized 32.9% is presented.

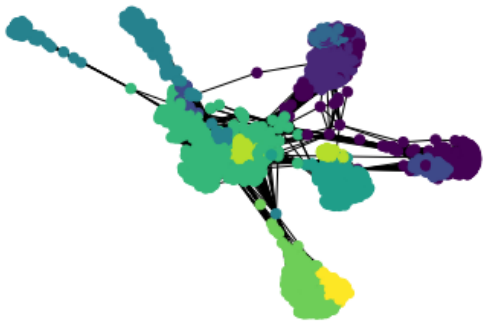


Fig. 1: Leskovec [9] graph plotted with networkx's spring layout and node colors representing community membership obtained by ComE BGMM+VI with $D = 4$ [10].

II. Friend Suggestions

Recommender Systems are eagerly researched in academia and widely deployed in real-world business applications. Most contemporary technology companies heavily rely on recommender systems to drive usage of their services and consumption of their content. Users, in turn, rely on recommender systems to find what they want and need and save searching time. State-of-the-art recommender systems provide a competitive advantage desperately needed by online services.

Companies heavily relying on recommender systems include YouTube, Amazon, Netflix, and many more [11].

Recommender systems are built on top of user-item interaction. Friend suggestions are a simpler type of recommender system. For friend suggestions, no distinction is made between users and items. The entity user is both the subject and object of recommendation. This results in a simple data model, which can be used for structural friend suggestions, as shown in Fig. 2.

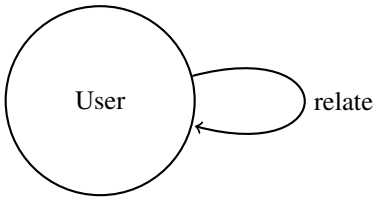


Fig. 2: Minimal data model supporting friend suggestions.

The two main methods used for recommender systems are collaborative filtering and content-based filtering. Collaborative filtering is based on the assumption that users like items similar to other items they like and items that are liked by other users with similar tastes [12, 13]. Content-based filtering considers user and item attributes, instead of solely interactions [11, 14].

In this paper, a method of generating friend suggestions based solely on graph data structure, and not on the node or edge attributes, is presented. The graph of users and friendships is embedded to a lower-dimensional space with the community embedding algorithm ComE BGMM+VI [8, 15, 16]. The approach can be considered a variant of collaborative filtering with the euclidean distance of user embeddings and user community membership as measures for user similarity. The proposed advantage of using such an approach is that community embeddings optimize first-, second-, and higher-order proximity between users in the node embedding.

A. Evaluation

To determine the effectivity of friend suggestions, the generated friend suggestions must be properly validated. In this paper, the top-N approach to evaluating recommender systems with hit-rate as the evaluation metric is chosen.

A user's top-N recommendations is a list of N items to be recommended to a specific user. To evaluate recommender systems with the top-N and hit-rate approach, initially, the dataset is split into train and test data. For each testing user, one relation is left out, according to the leave-one-out method and the model is trained on the remaining training dataset. Once the model is trained, a list of top-N recommendations is generated for each testing user. If the item corresponding to the user is in the user's top-N list, a hit is counted, otherwise, a miss is counted. The hit-rate is defined as the total number of hits divided by the number of testing users.

Utilizing the top-N approach with hit-rate to evaluate recommender systems is advantageous to evaluating recommender systems by a link prediction approach since the top-N approach is more realistic in comparison to actual recommender system use-cases. When you open Netflix, Amazon, YouTube, or Facebook friend suggestions, one or multiple top-N recommendation lists are generated and displayed. If you click on an item and buy, watch, or befriend, that is considered a hit, otherwise, a miss.

The hit-rate metric on top-N recommendations provides a realistic option of evaluating recommender systems and with the leave-one-out (LOO) strategy is used often in literature [17, 18, 19, 20].

III. Algorithm

The proposed algorithm for generating friend suggestions using community embeddings is detailed and evaluated on time complexity. The initially quadratic runtime of generating friend suggestions for all users is then reduced by a factor of K by utilizing a node's community membership. The two resulting algorithms are presented.

Algorithm 1 describes in pseudocode how recommendations are computed from node embeddings generated by ComE BGMM+VI. The terms friend suggestions and social recommendations are used interchangeably.

Algorithm 1 Top-N Social Recommendations based on Node Embeddings

Require: graph $G = (V, E)$, maximum number of communities K, number of walks γ, walk length ℓ, window size ζ, representation size D, negative context size m, parameters (α, β), number of recommendations N.
Ensure: Top-N recommendations for all nodes R.

$\Phi, \Phi', \Pi, (\Psi, \Sigma) \leftarrow ComE(G, K, \gamma, \ell, \zeta, D, m, \alpha, \beta)$
for $(v, v') \in E$ **do**
  $F_v \leftarrow F_v \cup \{v'\}$
  $F_{v'} \leftarrow F_{v'} \cup \{v\}$
end for
for $v \in V$ **do**
  $R_v \leftarrow SortedDict(size = N)$
  for $v' \in V \wedge v' \notin F_v \wedge v' \neq v$ **do**
    $d \leftarrow \|\phi_v - \phi_{v'}\|$
    $R_v[d] \leftarrow v'$
  end for
  $R_v \leftarrow R_v.values()$
end for

The function $ComE(G, K, \gamma, \ell, \zeta, D, m, \alpha, \beta)$ is the ComE BGMM+VI community embedding algorithm. It returns the node embedding Φ, context embedding Φ', community assignment Π, and community embedding (Ψ, Σ). ComE BGMM+VI is based on the ComE algorithm presented by Cavallari et al. in their paper Cavallari et al. [8] as algorithm 1 on page 381 of the publication's in-proceeding [8]. ComE BGMM+VI takes a Bayesian approach to community embedding by utilizing a Bayesian Gaussian mixture model (BGMM) and variational inference (VI) for community embedding and detection instead of a non-Bayesian Gaussian mixture model (GMM) with expectation maximization (EM) [16].

The runtime of ComE BGMM+VI is equivalent to the runtime of ComE in its big-O notation. ComE's time complexity is $O(|V|\,\gamma\ell + |V| + T_1(T_2|V|\,K + K + |E| + |V|\,\gamma\ell + |V|\,K))$, which is linear in time complexity to the graph size: $O(|V| + |E|)$ (line 1) [8]. All friends of each user are determined in $O(E)$ (lines 2-5). A sorted dictionary of the top-N friend suggestions for each user sorted ascending by distances between the node embeddings is computed in $O(|V|^2\,N))$ (lines 6-13). Top-N friend suggestions are generated for each user: $O(|V|)$. For each user, all users not befriended currently are considered ($O(|V|)$) and inserted into a sorted dictionary of length N ($O(N)$).

This brings the total time complexity of Algorithm 1 to $O(|V|\,\gamma\ell + |V| + T_1(T_2|V|\,K + K + |E| + |V|\,\gamma\ell + |V|\,K) + |E| + |V|^2\,N)$. We consider γ, ℓ, T_1, T_2, and N as constant, therefore the time complexity depends on the graph's size ($|V|$,$|E|$) and the number of communities K: $O(K + |V|\,K + |E| + |V|^2)$.

Algorithm 2 describes in pseudocode how recommendations are computed from ComE BGMM+VI node and community embeddings and community assignments. The advantage of also considers community membership, is that an improvement in time complexity can be obtained, by considering only users in the same community for friend suggestions.

Algorithm 2 Top-N Social Recommendations based on Node and Community Embeddings

Require: graph $G = (V, E)$, maximum number of communities K, number of walks γ, walk length ℓ, window size ζ, representation size D, negative context size m, parameters (α, β), number of recommendations N.

Ensure: Top-N recommendations for all nodes R.

1: $\Phi, \Phi', \Pi, (\Psi, \Sigma) \leftarrow ComE(G, K, \gamma, \ell, \zeta, D, m, \alpha, \beta)$
2: **for** $v \in V$ **do**
3: $\quad k \leftarrow \pi_v$
4: $\quad C_k \leftarrow C_k \cup \{v\}$
5: **end for**
6: **for** $(v, v') \in E$ **do**
7: $\quad F_v \leftarrow F_v \cup \{v'\}$
8: $\quad F_{v'} \leftarrow F_{v'} \cup \{v\}$
9: **end for**
10: **for** $v \in V$ **do**
11: $\quad R_v \leftarrow SortedDict(size = N)$
12: $\quad$ **for** $v' \in C_k \wedge v' \notin F_v \wedge v' \neq v$ **do**
13: $\quad\quad d \leftarrow \|\phi_v - \phi_{v'}\|$
14: $\quad\quad R_v[d] \leftarrow v'$
15: $\quad$ **end for**
16: $\quad R_v \leftarrow R_v.values()$
17: **end for**

Line 1, lines 6-9, and lines 10-17, except for line 12, are the same in Algorithm 2 as in Algorithm 1. All users are filtered into sets C_k, one for each community k in $O(|V|)$ (lines 2-5). Instead of considering the set of all non-friends users of size $|V|$, only users in the same community C_k of size $\frac{|V|}{K}$ are considered (line 12) in $O(\frac{|V|^2 N}{K})$.

TABLE I: Algorithm 1 and 2 time complexity.

K constant	Algorithm 1	$O(\|E\| + \|V\|^2)$
	Algorithm 2	$O(\|E\| + \|V\|^2)$
K scaling	Algorithm 1	$O(K + \|V\|\,K + \|E\| + \|V\|^2)$
	Algorithm 2	$O(K + \|V\|\,K + \|E\| + \frac{\|V\|^2}{K})$

This brings the total time complexity of Algorithm 2 to $O(|V|\,\gamma\ell + |V| + T_1(T_2|V|\,K + K + |E| + |V|\,\gamma\ell + |V|\,K) + |V| + |E| + \frac{|V|^2 N}{K})$. Again, we consider γ, ℓ, T_1, T_2, and N as constant, therefore the time complexity depends on the graph's size ($|V|$,$|E|$) and the number of communities K: $O(K + |V|\,K + |E| + \frac{|V|^2}{K})$.

The runtimes for Algorithm 1 and 2 when considering K as constant or scaling w.r.t the graph $G = (V, E)$ are shown in Table I. The number of friend suggestions N is considered constant w.r.t the graph $G = (V, E)$.

When the hyperparameter number of communities K is considered non-constant with respect to the graph G, Algorithm 2 reduces the quadratic runtime in comparison to Algorithm 2. K can therefore reduce the quadratic summand $|V|^2\,N$ by only considering users of the same community for friend suggestions. The reduction in time complexity through Algorithm 2 is only scalable when considering K to be scale with the input graph's size. This assumption can be made when considering communities as groups of friends: with ten times more users, these users form ten times more communities.

IV. Visual Example

The small Pakin [21] dataset will be used to visually underline the motivation for using node embedding and community memberships generated by ComE BGMM+VI for generating friend suggestions for person nodes on the karate club graph [21].

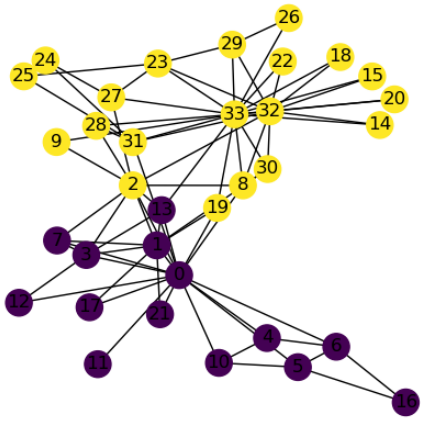


Fig. 3: Pakin [21] plotted with networkx's spring layout [10, 21].

TABLE II: Hyperparameters used for the Karateclub dataset.

parameter	notation	value
number_walks	γ	10
walk_length	ℓ	80
representation_size	D	2
num_workers		10
num_iter		3
reg_covar		0.00001
batch_size		50
window_size	ζ	10
negative	m	5
lr		0.025
alpha	α	0.1
beta	β	0.1
down_sampling		0.0
communities	K	5
weight_concentration_prior	Γ	10^{-5}

Fig. 3 shows the 34 nodes and 78 edges of the Pakin [21] dataset plotted as a graph with the Python network package networkx's spring layout.[10] Spring layout simulates a force-directed representation of the network, by treating edges as springs pulling nodes together and treating nodes as repelling objects, sometimes called an anti-gravity force.[10] The node classification, represented by node color, shown in Fig. 3 was obtained by running ComE with BGMM and VI with the hyperparameters presented in Table II.

As can be seen in Fig. 3, the two communities in the Pakin [21] dataset are correctly identified, despite the number of communities being initialized with $K = 5$. Thanks to the Bayesian approach to community modeling and optimization through variational inference, three unused communities are dropped, which leaves the prominent two communities present in the dataset. The node classification obtained reflects the node classification published in the 2017 original ComE paper by Cavallari et al..[8]

A. Training and Testing Split

Pakin [21] is split into training and testing parts. For evaluation, the leave-one-out (LOO) method is used, where for each testing user one edge is left out, then if the left out edge is in the user's top-N recommendations, a hit is counted, otherwise, a miss is recorded.

The three users with the highest degree are chosen as testing users: user 30, user 0, and user 32. For each testing user, a LOO-testing edge is chosen: (30, 23), (0, 1), and (32, 30). These three testing edges are spared for the testing dataset and omitted from the training graph.

B. Embedding

ComE BGMM+VI is run with the hyperparameters listed in Table II on the training dataset. The resulting node and community embeddings are visualized as node embeddings colored by community membership and community embeddings plotted as ellipses in Fig. 4.

C. Friend Suggesting

With embeddings generated, top-N friend suggestion recommendations are generated from the training node and community embeddings for the testing users. For each testing user, a ranked list of non-friend users from the same community is generated (Algorithm 2). A test user's friend suggestion list is ordered in ascending order by the euclidean distance between the node embeddings. Table III shows the test users' friend suggestion ranked lists.

The testing users' removed edges make it to the lists successfully; therefore, the removed friends are in the same community as the test users and can be recommended through friend suggestions.

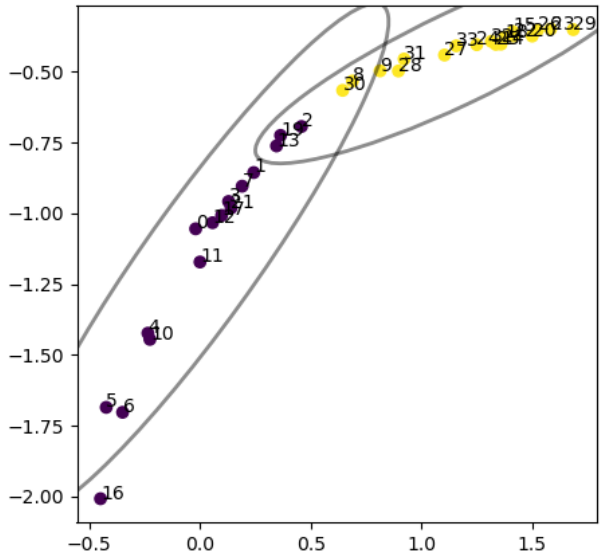


Fig. 4: Pakin [21] training dataset node and community embedding and node classification.

TABLE III: Testing users' friend suggestions ranked by node embedding distance.

User	Distance
24	0.137181
25	0.214332
23	0.543287

(a) user 33

User	Distance
1	0.322270
16	1.030336

(b) user 0

User	Distance
25	0.039816
24	0.045994
27	0.182484
26	0.255288
28	0.356683
9	0.501317
30	0.668306

(c) user 32

D. Evaluation

The top-N friend suggestions, generated for the testing users with Algorithm 2, are evaluated against the three removed test edges. The ranked lists of friend-suggestions shown in Fig. III is used to determine top-N friend suggestions for each testing user and evaluate the recommendations my the hit-rate metric. The LOO-testing edges are printed bold in Table III. The resulting top-N hit-rates are plotted against the number of suggestions N in Fig. 5.

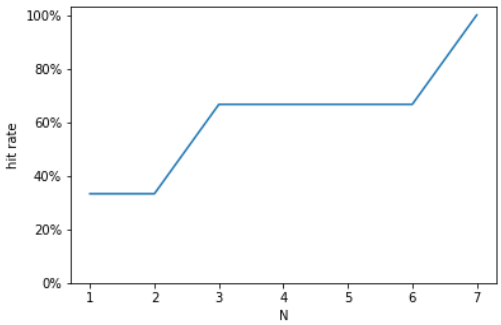


Fig. 5: Hit-rates from top-N recommendations on Pakin [21].

Fig. 5 shows promising results for using ComE to generate top-N friend suggestions on the small Pakin [21]. The achieved hit-rate starts at 33% for $N = 1$, reaches 67% at $N = 3$, and maxes out at 100% for $N = 7$; therefore, when presenting three friend suggestions to one test user, at least in 66% of cases a friend suggestion would be of interest.

V. Facebook Friend Suggestions

The recommender system based on community embeddings with the Bayesian Gaussian mixture model and variational inference is quantitatively evaluated. Friend suggestions are generated for the Leskovec [9] dataset [9, 22].

The method of generating friend suggestions using ComE BGMM+VI embeddings and the smallest-distance approach is applied to the Leskovec [9] graph. Both Algorithm 1 and Algorithm 2 are evaluated. The top-N recommendations are evaluated by the hit-rate metric.

A. Dataset

The Leskovec [9] dataset was sourced from the profile and network data of ten ego-networks (egonets) through a custom Facebook app by McAuley and Leskovec as part of their 2012 article McAuley and Leskovec [22].[22]

Fig. 6 shows an illustration of the ego-network of user u. User u has direct connections to v_i, called alters. Alters can be connected to each other and build circles of users, like families, high school friends, or college friends.

McAuley and Leskovec obtained profile and network data from ten Facebook ego-networks. The dataset contains 4039 users (nodes) and 88234 friendships (edges).

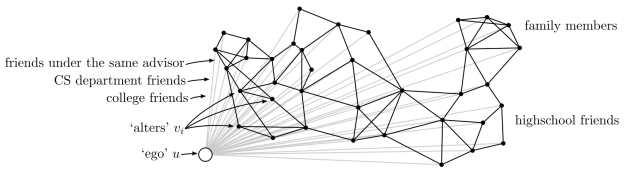


Fig. 6: Illustrative example of an ego-network [22].

TABLE IV: Hyperparameters used for the Leskovec [9] dataset.

parameter	notation	value
number_walks	γ	10
walk_length	ℓ	80
representation_size	D	$2, 4, 128$
num_workers		10
num_iter		3
com_n_init		10
reg_covar		0.00001
batch_size		50
window_size	ζ	10
negative	m	5
lr		0.025
alpha	α	0.1
beta	β	0.1
down_sampling		0.0
communities	K	10
weight_concentration_prior	Γ	10^{-5}

B. Training and Testing Split

The Leskovec [9] graph is split into training and testing parts. For evaluation, the leave-one-out (LOO) method is used, where for each testing user one edge is left out, then if the left out edge is in the user's top-N recommendations, a hit is counted, otherwise, a miss is recorded. Users with only one connection are removed from the dataset since no network information is added by them.

1169 of the 4039 users of the Leskovec [9] graph have at least 50 connections and are used for testing friend suggestions. The threshold of 50 friends for considering users for testing is set to make sure enough data is available to make sensable suggestions for testing users.

Now that the testing users are identified, the leave-one-out method is employed for obtaining one testing friendship per testing user. The friendship chosen to leave out for testing for each testing user is chosen randomly using NumPy's random choice.[23, 24]

C. Embedding

Community embeddings for the Leskovec [9] graph are generated by running ComE BGMM+VI with the hyperparameters listed in Table IV.

As is shown in Table IV, the embedding's representation size D is varied along three values. A representation size $D \in [2, 4, 128]$ is chosen to determine the effect of the embedding's dimensionality on friend suggestions. It is hypothesized, that two dimensions are not sufficient to capture the structure of the Leskovec [9] graph, due to its size, therefore higher dimensional embeddings are evaluated as well.

Fig. 7, Fig. 8, and Fig. 9 show the node embeddings generated by ComE BGMM+VI after one, two, and three iterations. The nodes are colored by community membership from ComE's community embedding and detection. The 4- and 128-dimensional node and community embeddings cannot effectively be plotted in two-dimensional print. Instead, in Fig. 8 and Fig. 9 the two first dimensions of the node and community embeddings are plotted to gain a narrow perspective on the resulting node embeddings.

Fig. 10 shows the graph spring plotted with networkx and nodes colored by community membership after the lastest iteration of ComE.[10] The spring plotted graph structure on Fig. 10 suggest, that with embedding dimension $D = 4$, more fitting communities are detected than for $D = 2$ and $D = 128$.

Fig. 7 shows node and community embeddings with representation size $D = 2$. The limiting two-dimensional space does not allow ComE to represent the Leskovec [9] graph's structure sufficiently to utilize the $K = 10$ components of the Gaussian mixture model. Only three components are used after the last of three iterations.

While Fig. 8 and Fig. 9 show only the first two dimensions, of the node embedding of 4 or 128 dimensionality respectively, the general structure of communities is visible. The development of the node embeddings over the three iterations shows the manifestation of the first-, second-, and higher-order proximity by ComE's closed loop. The node embeddings are densified in their communities and spread out in regards to other communities. All $K = 10$ components of the Gaussian mixture model are used for $D = 4$ and $D = 128$.

The nodes are distributed over the $K = 10$ communities as shown in the histograms in Fig. 11. Fig. 11a shows that with representation size $D = 2$, ComE BGMM+VI drops 6 communites, effecitvelly using only 4 communities in the final embedding, because the two-dimensional node embeddings are not expressive enough to support more communities. Fig. 11b and Fig. 11c show that with representation sizes $D = 4$ and $D = 128$ respectivelly, all $K = 10$ communities are used.

The tradeoff between the representation size of the embedding D and the embedding expressiveness is demonstrated. When choosing a too low-dimensional embedding, the embedding might not be expressive enough to capture the communities and reduces the quality of the community embedding and detection. When choosing a too high-dimensional embedding, the node embeddings get so expressive that BGMM+VI is not able to drop unused communities because all communities are used in one of the many dimensions.

D. Friend Suggestions

With node and community embeddings and community assignments generated for $D \in [2, 4, 128]$ and $K = 10$ with training data in the previous section, top-N friend suggestion recommendations are generated from the training node and community embeddings with Algorithm 1 and Algorithm 2.

The shortest-embeddings-distance approach is employed, recommending friends to users that have the shortest distance node embeddings over all users (Algorithm 1) or only over users in the same community (Algorithm 2). Users who are already connected are not recommended again.

VI. Evaluation

The charts in Fig. 12 show the hit-rates for top-N recommendations for testing users.

Two hit-rate lines are shown for each embedding representation size $D \in [2, 4, 128]$ on N-axis. The blue line shows the hit-rates when generating friend suggestions from all users (Algorithm 1). The orange line shows the hit-rate when generating recommendations from users in the same community (Algorithm 2). For lower-dimensional embeddings with $D \in [2, 4]$, the hit-rates for Algorithm 1 and Algorithm 2 largely overlap.

Table 13 shows readings of hit-rates over the algorithms and embedding dimensionality for top-5, top-10, and top-20 recommendations:

Hit-rates based on top-5, top-10, and top-20 recommendations are chosen since higher N are not practical for recommendations.

Presenting only the top-10 friend suggestions to users, a hit-rate of 43.6% when considering all users, or 32.9%, when considering only users from the same community, is achieved. A top-5 hit-rate of 31.3% (all users) and 24.1% (same community) is achieved, which can be considered viable for business use-cases, especially for only showing 5 recommendations to users. A top-20 hit-rate of 58.8% (all users) and 41.8% (same community) adds to the presented results.

VII. Conclusion

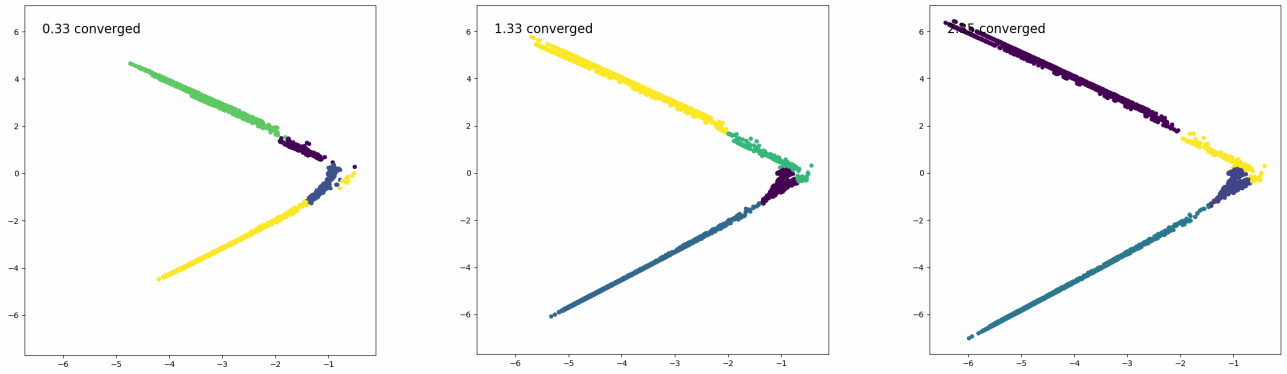
The proposed shortest-embedding-distance friend suggestion recommender system based on embeddings generated with the community embeddings algorithm ComE BGMM+VI shows promising results. A top-10 recommendations hit-rate of 43.6% when considering all users (Algorithm 1), and over 32.9% when considering users in the same community (Algorithm 2) is achieved for Facebook friend suggestions.

A. Future Work

This work explores generating friend suggestions from node and community embeddings generated by ComE. Three values of the representation size D were used in experimentation. Experimenting with more, different combinations of hyperparameters, and especially varying the number of communities K might be of interest. Since in this paper, friend suggestions were explored, it is left for future work to explore item-based recommendations using embeddings. More community-centric recommendations, which deeply consider the community embeddings in generating recommendations, are left for future work.

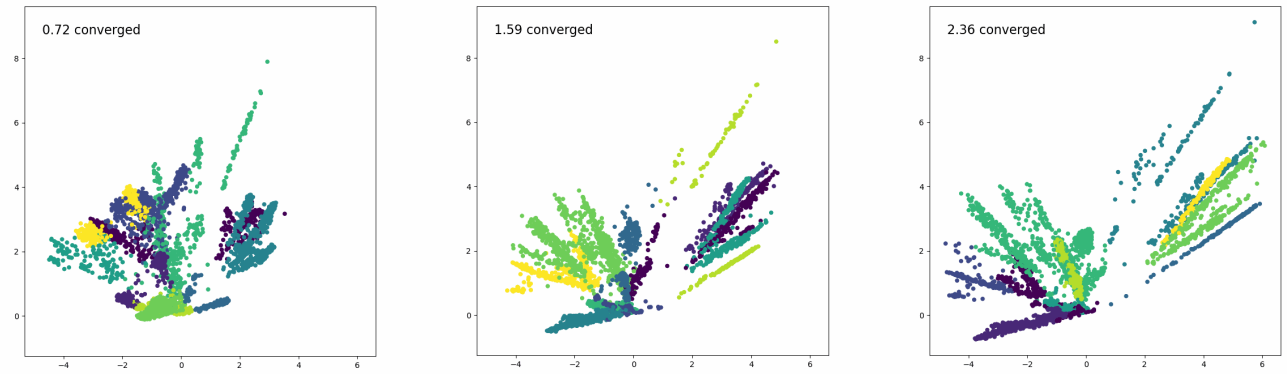
References

[1] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, p. 78–94, 7 2018. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2018.03.022



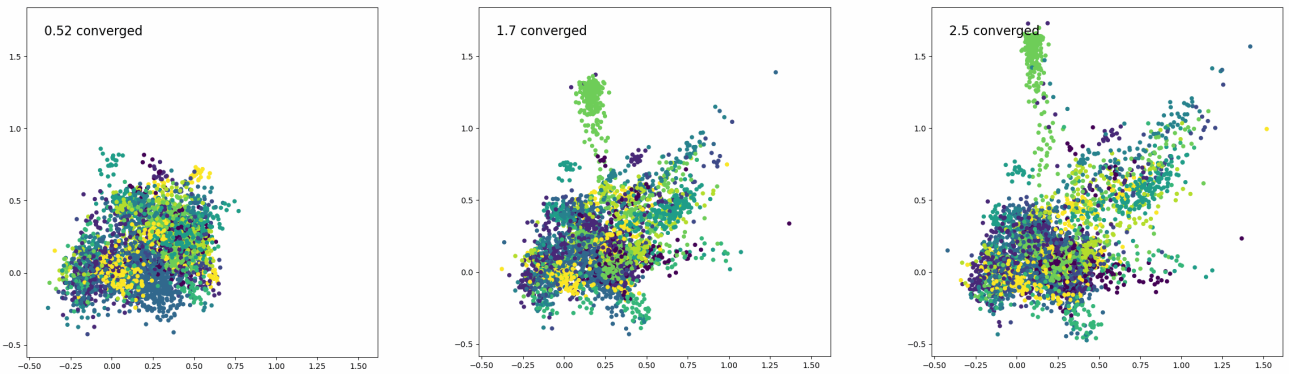
(a) 1st Iteration (b) 2nd Iteration (c) 3rd Iteration

Fig. 7: Node embeddings segment with representation size $D = 2$.



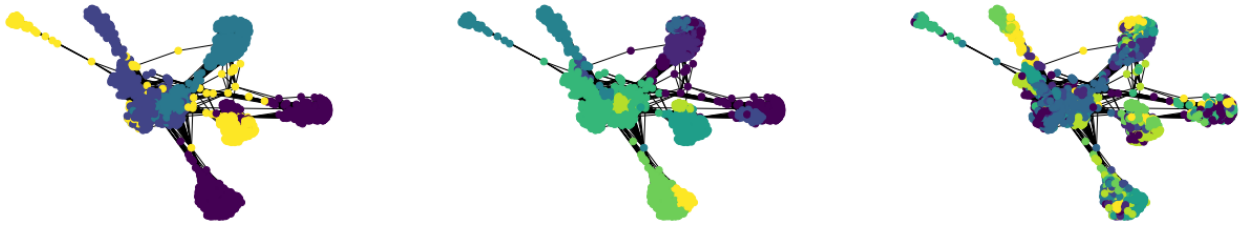
(a) 1st Iteration (b) 2nd Iteration (c) 3rd Iteration

Fig. 8: Node embeddings segment with representation size $D = 4$.



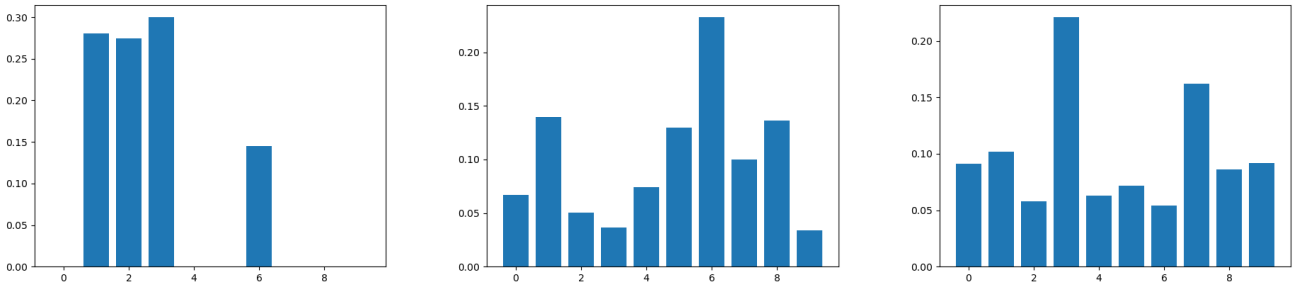
(a) 1st Iteration (b) 2nd Iteration (c) 3rd Iteration

Fig. 9: Node embeddings segment with representation size $D = 128$.



(a) $D = 2$ (b) $D = 4$ (c) $D = 128$

Fig. 10: Spring plotted graphs with nodes colored by community membership after last ComE BGMM+VI iteration.



(a) $D = 2$ (b) $D = 4$ (c) $D = 128$

Fig. 11: Weights of communities with $K = 10$ and $D \in [2, 4, 128]$.

[2] N. Polatidis and C. K. Georgiadis, "Recommender systems: The importance of personalization in e-business environments," *IJEEI*, vol. 4, pp. 32–46, 2013.

[3] E. Palumbo, G. Rizzo, R. Troncy, E. Baralis, M. Osella, and E. Ferro, "An empirical comparison of knowledge graph embeddings for item recommendation," in *DL4KGS@ESWC*, 2018.

[4] ——, "Knowledge graph embeddings with node2vec for item recommendation," in *ESWC*, 2018.

[5] L. Grad-Gyenge, A. Kiss, and P. Filzmoser, "Graph embedding based recommendation techniques on the knowledge graph," 07 2017, pp. 354–359.

[6] V. Sathish, T. Mehrotra, S. Dhinwa, and B. Das, "Graph embedding based hybrid social recommendation system," *ArXiv*, vol. abs/1908.09454, 2019.

[7] P. Godec. (2018) Graph embeddings — the summary. [Online]. Available: https://towardsdatascience.com/graph-embeddings-the-summary-cc6075aba007

[8] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ser. CIKM '17. Association for Computing Machinery, 2017, p. 377–386. [Online]. Available: https://doi.org/10.1145/3132847.3132925

[9] J. Leskovec. (2012) Social circles: Facebook. [Online]. Available: https://snap.stanford.edu/data/egonets-Facebook.html

[10] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[11] B. Rocca, "Introduction to recommender systems," *Towards Data Science*. [Online]. Available: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada

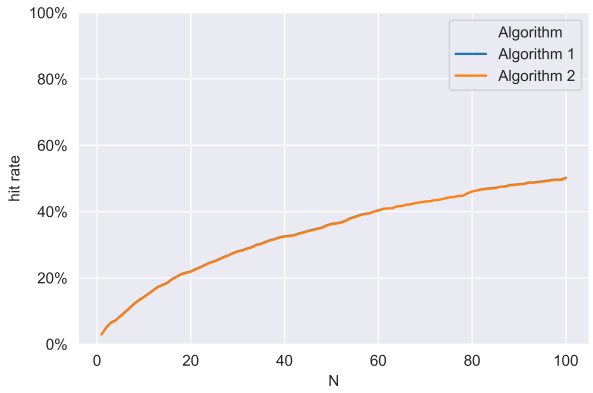
[12] P. Grover, "Various implementations of collaborative filtering," *Towards Data Science*. [Online]. Available: https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

[13] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. Artif. Intell.*, vol. 2009, pp. 421 425:1–421 425:19, 2009.

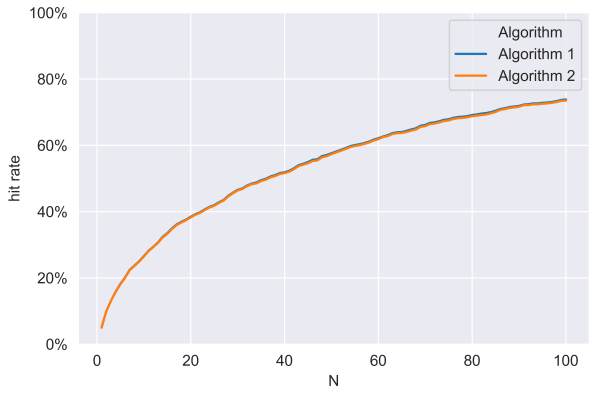
[14] P. Lops, M. de Gemmis, and G. Semeraro, *Content-based Recommender Systems: State of the Art and Trends*, 01 2011, pp. 73–105.

[15] S. Cavallari, E. Cambria, H. Cai, K. C. Chang, and V. W. Zheng, "Embedding both finite and infinite communities on graphs [application notes]," *IEEE Computational Intelligence Magazine*, vol. 14, no. 3, pp. 39–50, 2019.

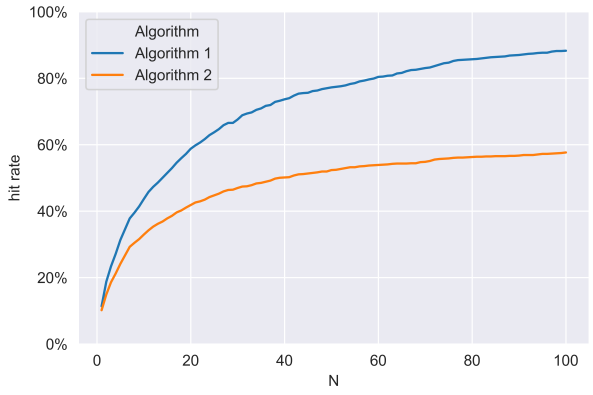
[16] A. Begehr. (2020) abegehr/ComE_BGMM. [Online].



(a) $D = 2$ on linear N-axis.



(b) $D = 4$ on linear N-axis.



(c) $D = 128$ on linear N-axis.

Fig. 12: Hit-rates of Leskovec [9] graph's top-N friend suggestions on N-axis for $D \in [2, 4, 128]$. Hit-rate lines for Algorithm 1 and Algorithm 2 overlap for $D = 2$ and $D = 4$.

Fig. 13: Hit-rates of Leskovec [9] graph's top-N friend suggestions for algorithms 1 and 2, $N \in [10, 20]$, and $D \in [2, 4, 128]$.

		top-5	top-10	top-20
Algorithm 1	$D = 2$	8.5%	14.2%	21.9%
	$D = 4$	18.1%	26.5%	38.4%
	$D = 128$	**31.3%**	**43.6%**	**58.8%**
Algorithm 2	$D = 2$	8.4%	14.1%	21.8%
	$D = 4$	18.0%	26.4%	38.2%
	$D = 128$	**24.1%**	**32.9%**	**41.8%**

Available: https://github.com/abegehr/ComE_BGMM

[17] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," 01 2010, pp. 39–46.

[18] E. Palumbo, G. Rizzo, and R. Troncy, "Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, ser. RecSys '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 32–36. [Online]. Available: https://doi.org/10.1145/3109859.3109889

[19] Z. Zhao, M. Zhu, Y. Sheng, and J. Wang, "A top-n-balanced sequential recommendation based on recurrent network," *IEICE Trans. Inf. Syst.*, vol. 102-D, pp. 737–744, 2019.

[20] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW '17. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017, p. 173–182. [Online]. Available: https://doi.org/10.1145/3038912.3052569

[21] S. Pakin. (1977) Zachary's Karate Club graph. [Online]. Available: https://networkx.github.io/documentation/stable/auto_examples/graph

[22] J. McAuley and J. Leskovec, "Learning to discover social circles in ego networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. Curran Associates Inc., 2012, p. 539–547.

[23] T. E. Oliphant, *A Guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[24] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, p. 22, 2011.